ARL-TN-0764 • June 2016

US Army Research Laboratory

Mission Driven Scene Understanding: Dynamic Environments

by Arnold Tunick

NOTICES

Disclaimers

ARL-TN-0764 ● JUNE 2016

US Army Research Laboratory

Mission Driven Scene Understanding: Dynamic Environments

by Arnold Tunick
Computational and Information Sciences Directorate, ARL

REPORT DOCUMENTATION PAGE			*Form Approved* *OMB No. 0704-0188*

1. REPORT DATE *(DD-MM-YYYY)*	**2. REPORT TYPE**	**3. DATES COVERED (From - To)**
June 2016	Technical Note	10/2015 – 06/2016

4. TITLE AND SUBTITLE Mission Driven Scene Understanding: Dynamic Environments	**5a. CONTRACT NUMBER**
	5b. GRANT NUMBER
	5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S) Arnold Tunick	**5d. PROJECT NUMBER**
	5e. TASK NUMBER
	5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) US Army Research Laboratory ATTN: RDRL-CII-A 2800 Powder Mill Road Adelphi, MD 20783-1138	**8. PERFORMING ORGANIZATION REPORT NUMBER** ARL-TN-0764
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)	**10. SPONSOR/MONITOR'S ACRONYM(S)**
	11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT

Approved for public release; distribution unlimited.

13. SUPPLEMENTARY NOTES

14. ABSTRACT

Knowledge of how time and space changing environmental conditions cause changes in the context of images is necessary for scene understanding. Such dynamic environmental conditions (e.g., changing illumination, precipitation, and vegetation) can modify saliency and context, obscure features, and degrade object recognition. Here, *context* means more than the typically referenced attributes, content, or composition of an outdoor scene. For Army applications, scene understanding needs to be viewed in the context of providing optimal value to the Army mission. Then, for example, helpful image cues that relate to mission activities may include time of day, current and future weather conditions, visibility, terrain, and scene location. In this report, we outline progress toward implementing our *mission driven scene understanding* approach to advance the value of Army autonomous intelligent systems. We describe the proof-of-principle installation, setup, and testing of a convolutional neural network (CNN) program developed in Python and all its required software dependencies. While we found that the CNN was able to determine the correct class labels for images taken from the training data set, the validation process did not appear to provide optimal results for images not previously seen. Thus, we recommend performing additional trials and analysis to better determine the feasibility of using the CNN to augment our approach.

15. SUBJECT TERMS

computer vision, context, saliency, visibility, illumination, convolutional neural network

16. SECURITY CLASSIFICATION OF:			**17. LIMITATION OF ABSTRACT**	**18. NUMBER OF PAGES**	**19a. NAME OF RESPONSIBLE PERSON** Arnold Tunick
a. REPORT	**b. ABSTRACT**	**c. THIS PAGE**			**19b. TELEPHONE NUMBER (Include area code)**
Unclassified	Unclassified	Unclassified	UU	26	(301) 394-1233

Standard Form 298 (Rev. 8/98)
Prescribed by ANSI Std. Z39.18

Contents

List of Figures **v**

List of Tables **v**

Acknowledgments **vi**

1. Introduction **1**

2. Prerequisite Software Installation **2**

2.1 GIT for Windows 2

2.2 Visual Studio Community 2013 3

2.3 Windows Software Development Kit for Windows 10 3

2.4 CUDA v7.5 3

2.5 TDM-GCC 3

2.6 Scientific Python v2.7.9.4 3

3. Installing Theano V0.8.0 **4**

3.1 Configuration of Paths 4

3.2 Test the Configuration of Paths 5

3.3 Link Library for GCC 5

3.4 Setup/Install Theano 5

3.5 Test Theano: CPU 5

3.6 Test Theano: GPU 6

3.7 Additional Theano Test 6

4. AlexNet CNN Implementation with Theano **7**

4.1 PIP 7

4.2 Pycuda 7

4.3 Hickle 8

4.4 Pylearn2 8

4.5 Theano-Alexnet 8

4.6 Prepare and Preprocess ImageNet Data 8

4.6.1 Set Configurations Paths for AlexNet 9

4.6.2 Preprocessed ImageNet Data for Theano-AlexNet 10

4.7 Train Theano-AlexNet 11

5. Summary and Conclusions 13

References 14

Distribution List 18

List of Figures

Fig. 1 A few example images from a subset of the ImageNet ILSVRC2012 data used for the short trial training of the Theano-AlexNet code that illustrate time- and space-varying environmental conditions in outdoor scenes, such as variations in illumination, vegetation, terrain, and visibility 11

List of Tables

Table 1 Folders generated in C:\SciSoft\Git\theano_alexnet\scratch\ilsvrc12\ 10

Table 2 Files generated in C:\SciSoft\Git\theano_alexnet\scratch\ilsvrc12\labels 10

Table 3 Files generated in C:\SciSoft\Git\theano_alexnet\scratch\ilsvrc12\misc 10

Table 4 Building the model..12

Table 5 CNN training and validation results: 20,000 iterations 12

Acknowledgments

I thank RE Meyers, P David, G Warnell, B Byrne, C Karan, and S Gutstein for helpful discussions. This research was supported by the US Army Research Laboratory.

1. Introduction

Rapid and robust scene understanding is a critically important goal for the development of Army autonomous intelligent systems.[1] For outdoor natural scenes, autonomous intelligent systems will need to quickly discern the depth of view, navigability, exposure or concealment (as it relates to object searching), and transience, that is, the rate at which elements of the scene or its environment are changing in space and time.[2,3] In this regard, saliency estimation has been helpful to computationally identify elements in a scene that immediately capture the visual attention of an observer.[4,5] Several recent papers have discussed concepts associated with visual saliency to enhance automated navigation and scene exploration.[6–8] Note, however, that the most active or salient object(s) in a scene, by this definition,[4,5,9] may not represent the most important or meaningful feature(s) of the scene in the context of the Army mission.[10] In other words, visual saliency also can be used to highlight key image cues that relate to Army mission activities.[10] For example, an automated vision system may readily detect changes in the ground surface as a new or different object in the field of view; however, recognizing the physical characteristics of the new surface (e.g., shallow or deep water, thick, thin, or melting ice, freezing rain, snow, mud, quicksand, and so on) and observing any changes in the context of the image may be critically important.[10–12] Characterizing interactions between objects and the environment also can contribute to physical scene understanding.[13–16]

Furthermore, knowledge of how time and space changing environmental conditions cause changes in the context of images is necessary for scene understanding.[10–12] Here, *context* means more than the typically referenced attributes, content, or composition of an outdoor scene.[17–19] For Army applications, scene understanding needs to be viewed in the context of providing optimal value to the Army mission. Then, for example, helpful image cues that relate to mission activities may include time of day, current and future weather conditions, visibility, terrain, and scene location. For instance, changing weather elements on the battlefield can alter terrain features and trafficability; low visibility can impede reconnaissance and target acquisition or alternately conceal friendly forces maneuvers and activities; and wind speed and direction can favor upwind forces in nuclear, biological, and chemical attacks or decrease the effectiveness of downwind forces due blowing dust, smoke, sand, rain, sleet, or snow.[20–25] In fact, any image cue that can potentially help the mission should not be overlooked, since it will aid scene understanding in the context of the Army mission. Consequently, due to bandwidth and/or operations constraints, there will be a need for metrics to prioritize image cuing that relate to mission activities. Thus, our *mission driven scene understanding* approach is designed to optimize mission success.

Many of the current methods for scene understanding, like those that generate image descriptions via automated semantic labeling[26] or visual scene classification,[27] are only beginning to address changing environmental conditions (e.g., with regard to identifying changes in terrain characteristics to enhance autonomous navigation processes).[28] Yet, considering context changes (e.g., due to a changing environment) can pose serious challenges for computer vision processes, such as those associated with place recognition, navigation, road/terrain detection, and scene exploration.[29–33] This is because rain, snow, and fog weather events, as well as smoke, haze, or other changes in lighting and visibility can modify saliency and context of an outdoor scene, obscure features, and significantly degrade object recognition.[34–37] Naturally, scene-depicted environmental conditions can vary with time of day, season, and location.[38]

In this report, we outline progress toward implementing our *mission driven scene understanding* approach to advance the value of Army autonomous intelligent systems and support the Army mission in complex and changing battlefield environments. We describe the proof-of-principle installation, setup, and testing of a convolutional neural network (CNN) program developed in Python and all its required software dependencies.[39–42] Here, we suggest that the CNN could be tested initially with simple single-object images and later on with more–complicated scenes, such as those illustrating changes in illumination, vegetation, terrain, and visibility.

2. Prerequisite Software Installation

In this section, we outline the prerequisite software installations to implement the Theano program code[39,40] on a Windows 10 notebook computer. Here, Theano is a Python library that facilitates the efficient evaluation of mathematical expressions involving multidimensional arrays. Alternately, an online overview for installing Theano on Windows can be found at https: // deeplearning.net/software/theano/ install_windows.html#install-windows .

2.1 GIT for Windows

To access the GitHub software repository, download the 64-bit version of GIT from https: // github.com/git-for-windows/git/releases/tag/v2.7.1.windows.2 and extract the files into the folder C:\SciSoft\Git.

2.2 Visual Studio Community 2013

To access a C++ integrated development environment with 64-bit compilers, download Visual Studio Community 2013 from https: // www visualstudio.com/en-us/news/vs2013-community-vs.aspx. Installation and setup for this software is self-explanatory, although one does need to add the following 3 folders to the path:

1. C:\Program Files (x86)\Microsoft Visual Studio 12.0\VC\bin\amd64
2. C:\Program Files (x86)\Microsoft Visual Studio 12.0\VC\lib\amd64
3. C:\Program Files (x86)\Microsoft Visual Studio 12.0\VC\include

2.3 Windows Software Development Kit for Windows 10

In addition to Visual Studio 12.0, download the Windows software development kit for Windows 10 from https: // dev.windows.com/en-us/downloads/windows-10-sdk and extract the files into the folder C:\Program Files (x86)\Microsoft Visual Studio 12.0\VSSDK. The VSSDK folder should also be added to the path.

2.4 CUDA v7.5

To provide a development environment for C++ programs implementing graphics processing unit (GPU)-accelerated applications, download CUDA v7.5 from https: // developer.nvidia.com/cuda-toolkit. This software installation will require that a supported version Microsoft Visual Studio be found on the computer. If not completed automatically, the path can be updated to include the following 2 folders:

1. C:\Program Files\NVIDIA GPU Computing Toolkit\CUDA\v7.5\libnvvp
2. C:\Program Files\NVIDIA GPU Computing Toolkit\CUDA\v7.5\bin

2.5 TDM-GCC

The Theano code compiler requires TDM-GCC installation for either 32- or 64-bit platforms. Therefore, one needs to download the 64-bit version TDM-GCC software from http: // tdm-gcc.tdragon.net/ and extract the files into the folder C:\SciSoft\TDM-GCC-64.

2.6 Scientific Python v2.7.9.4

To provide the necessary Python components for both Theano[39,40] and the CNN AlexNet[41,42] and for all of their programs' software dependencies, such as numpy, hickle, pycuda, pylearn2, and zeromq, download and install the 64-bit version

Python v2.7.9.4 from https: // sourceforge.net/projects/winpython/files/ WinPython_2.7/2.7.9.4/ and extract the files into the folder C:\SciSoft\WinPython-64bit-2.7.9.4.

3. Installing Theano V0.8.0

To provide the mathematical framework within which the CNN AlexNet compiles, download the most current 64-bit version of Theano (v0.8.0) from https: // github.com/Theano/Theano and extract the files into the folder C:\SciSoft\Git\theano. Alternately, one can download and install the Theano files from a command window by typing the following at the prompt:

- C:\SciSoft\git> git clone https: // github.com/Theano/Theano.git

3.1 Configuration of Paths

To configure the system path for Python and Visual Studio, save following shell script as C:\SciSoft\env.bat:

```
REM configuration of paths
set VSFORPYTHON="C:\Program Files (x86)\Microsoft Visual Studio
12.0"
set SCISOFT=%~dp0
REM add tdm gcc stuff
set PATH=%SCISOFT%TDM-GCC-64\bin;%SCISOFT%TDM-GCC-64\x86_64-w64-
mingw32\bin;%PATH%
REM add winpython stuff
CALL %SCISOFT%WinPython-64bit-2.7.9.4\scripts\env.bat
REM configure path for msvc compilers
CALL %VSFORPYTHON%\vcvarsall.bat amd64
REM return a shell
cmd.exe /k
```

Note here that the file vcvarsall.bat, which is called within the env.bat shell script, should contain the following path information:

```
:amd64
echo Setting environment for using Microsoft Visual Studio 2013
x64 tools.
set VCINSTALLDIR=%~dp0VC\
REM set WindowsSdkDir=%~dp0WinSDK\
set WindowsSdkDir=%~dp0VSSDK\
if not exist "%VCINSTALLDIR%bin\amd64\cl.exe" goto missing
set PATH=%VCINSTALLDIR%Bin\amd64;%WindowsSdkDir%VisualStudioInteg
ration\Tools\Bin;%PATH%
set INCLUDE=%VCINSTALLDIR%Include;%WindowsSdkDir%VisualStudioInte
gration\Common\Inc;%INCLUDE%
set LIB=%VCINSTALLDIR%Lib\amd64;%WindowsSdkDir%VisualStudioIntegr
ation\Common\Lib\x64;%LIB%
set LIBPATH=%VCINSTALLDIR%Lib\amd64;%WindowsSdkDir%VisualStudio
```

```
Integration\Common\Lib\x64;%LIBPATH%
goto :eof
```

3.2 Test the Configuration of Paths

To test the path configuration, open the Python shell in a command window by typing C:\SciSoft\env.bat and then verify that the following programs are found by typing these lines at the prompt:

- C:\SciSoft> where gcc
- C:\SciSoft> where gendef
- C:\SciSoft> where cl
- C:\SciSoft> where nvcc

3.3 Link Library for GCC

To create a link library for GCC, open the Python shell in a command window by typing C:\SciSoft\env.bat and then type the following at the command window prompt:

- C:\SciSoft> gendef WinPython-64bit-2.7.9.4\python-2.7.9.amd64\ python27.dll
- C:\SciSoft> dlltool -dllname python27.dll -def python27.def -output-lib WinPython-64bit-2.7.9.4\python- 2.7.9.amd64\libs\libpython27.a

3.4 Setup/Install Theano

Finally, to set up and install Theano, open the Python shell in a command window by typing C:\SciSoft\env.bat and then type the following at the prompt:

- C:\SciSoft\Git\Theano> python setup.py develop

3.5 Test Theano: CPU

To test whether Theano works and is able to compile code for central processing unit (CPU) execution, create the following test file (e.g., filename = test.py):

```
import numpy as np
import time
import theano
A = np.random.rand(1000,10000).astype(theano.config.floatX)
B = np.random.rand(10000,1000).astype(theano.config.floatX)
np_start = time.time()
AB = A.dot(B)
np_end = time.time()
```

```
X,Y = theano.tensor.matrices('XY')
mf = theano.function([X,Y],X.dot(Y))
t_start = time.time()
tAB = mf(A,B)
t_end = time.time()
print("NP time: %f[s], theano time: %f[s] %(np_end-np_start,
t_end-t_start))
```

Then open the Python shell in a command window and type the following at the prompt:

- C:\SciSoft\Git\Theano> python test.py

The following is the example result:

```
NP time: 1.480863[s], theano time: 1.475381[s]
```

3.6 Test Theano: GPU

To test whether Theano works and is able to compile code for GPU execution, create the file .theanorc.txt in C:\SciSoft\WinPython-64bit-2.7.9.4\settings as follows:

```
[global]
device = gpu
REM device = cpu
floatX = float32
[nvcc]
flags=-LC:\SciSoft\WinPython-64bit-2.7.9.4\python\2.7.9.amd64\
libs
compiler_bindir=C:\Program Files (x86)\Microsoft Visual Studio
12.0\VC\bin
```

```
Then, rerun the test.py file shown in Section 3.5.
```

3.7 Additional Theano Test

As an additional test of the Theano code, open the Python shell in a command window and type the following at the prompt:

- C:\SciSoft\Git\Theano>python C:\SciSoft\Git\Theano\bin\theano-nose -batch=3000

The following is the example result:

```
####################
# COLLECTING TESTS #
###################################
# RUNNING TESTS IN BATCHES OF 3000 #
###################################
100% done in 604.919s (failed: 0)
####################
# ALL TESTS PASSED #
####################
```

4. AlexNet CNN Implementation with Theano

In this section, we outline all of the prerequisite software installations to implement the AlexNet CNN[42] program code within Theano on a Windows 10 notebook computer. Alternately, an online overview of configuring the paths for the AlexNet CNN,[42] preprocessing image data, and running the Python code can be found at https: // github.com/uoguelph-mlrg/theano_alexnet.

4.1 PIP

An alternate way to install the Python site packages (e.g., pycuda) is to download get-pip.py from https: // pip.pypa.io/en/stable/installing/, which can be extracted into the folder C:\SciSoft\WinPython-64bit-2.7.9.4\python-2.7.9.amd64\Scripts. Then to install PIP, open the Python shell C:\SciSoft\env.bat in a command window and type the following:

- C:\SciSoft\WinPython-64bit-2.7.9.4\python-2.7.9.amd64\
  Scripts> python get-pip.py

4.2 Pycuda

To install this dependent Python site package, download the file “pycuda-2015.1.3+cuda7518-cp27-none-win_amd64.whl” from http: // www lfd.uci.edu/ ~gohlke/pythonlibs/#pycuda and copy it to the folder C:\SciSoft\WinPython-64bit-2.7.9.4\settings\pipwin\. Then to install pycuda, open the Python shell in C:\SciSoft\env.bat and then at the command prompt type the following:

- C:\SciSoft>pip install C:\SciSoft\WinPython-64bit-2.7.9.4\settings\
  pipwin\pycuda-2015.1.3+cuda7518-cp27- none-win_amd64.whl

It is necessary to install several required C++ libraries prior to completing the steps for installing pycuda, as outlined above. Here, one needs to download boost_1_59_0-msvc-12.0-64.exe from https: // sourceforge.net/projects/boost/files/ boost-binaries/ and then double click on the file to install boost in the folder C:\local\boost_1_59_0.

4.3 Hickle

To install this dependent Python site package, download hickle from https: // github.com/telegraphic/hickle and then open the Python shell in C:\SciSoft\env.bat and then type the following at the command window prompt:

- C:\SciSoft> cd C:\SciSoft\WinPython-64bit-2.7.9.4\python-2.7.9.amd64\Lib\site-packages\hickle
- C:\SciSoft\WinPython-64bit-2.7.9.4\python-2.7.9.amd64\Lib\site-packages\hickle> python setup.py install

4.4 Pylearn2

To install this dependent Python site package, download pylearn2 from https: // github.com/lisa-lab/pylearn2 and then open the Python shell in C:\SciSoft\env.bat and then type the following at the command window prompt:

- C:\SciSoft> cd C:\SciSoft\WinPython-64bit-2.7.9.4\python-2.7.9.amd64\Lib\site-packages\pylearn2
- C:\SciSoft\WinPython-64bit-2.7.9.4\python-2.7.9.amd64\Lib\site-packages\pylearn2> python setup.py install

4.5 Theano-Alexnet

Download Theano-Alexnet from https: // github.com/uoguelph-mlrg/ theano_alexnet and extract files into the folder: C:\SciSoft\Git\theano_alexnet\.

4.6 Prepare and Preprocess ImageNet Data

To prepare and preprocess ImageNet data,[43] register and download the ImageNet Large Scale Visual Recognition Challenge 2012 (ILSVRC2012) image data .tar files and the 2014 development kit from http: // www image-net.org into the following 3 folders:

- C:\SciSoft\Git\theano_alexnet\mnt\data\datasets\ilsvrc_2014\ILSVRC2012_DET_train

- C:\SciSoft\Git\theano_alexnet\mnt\data\datasets\ilsvrc_2014\ILSVRC2012_DET_val
- C:\SciSoft\Git\theano_alexnet\mnt\data\datasets\ilsvrc_2014\ILSVRC2014_devkit

After downloading the image data, open the Python shell C:\SciSoft\env.bat and in the command window run the script C:\SciSoft\Git\theano_alexnet\preprocessing\generate_data.sh, which will call 3 Python scripts. This program runs for about 1–2 days. Alternately, for a short trial of the AlexNet code, run the script C:\SciSoft\Git\theano_alexnet\preprocessing\generate_toy_data.sh, which takes about 10 min.

4.6.1 Set Configurations Paths for AlexNet

Prior to preprocessing the image data, modify the path information in the file C:\SciSoft\Git\theano_alexnet\preprocessing\path.yaml as follows and be sure to make similar path annotations in the file C:\SciSoft\Git\theano_alexnet\spec_1gpu.yaml:

```
# dir that contains folders like n01440764, n01443537, ...
train_img_dir:'C:\SciSoft\Git\theano_alexnet\mnt\data\datasets\
ilsvrc_2014\ILSVRC2012_DET_train\'
# dir that contains ILSVRC2012_val_00000001~50000.JPEG
val_img_dir:'C:\SciSoft\Git\theano_alexnet\mnt\data\datasets\
ilsvrc_2014\ILSVRC2012_DET_val\'
# dir to store all the preprocessed files
tar_root_dir:'C:\SciSoft\Git\theano_alexnet\scratch\ilsvrc12'
# dir to store training batches
tar_train_dir:'C:\SciSoft\Git\theano_alexnet\scratch\ilsvrc12\
train_hkl'
# dir to store validation batches
tar_val_dir:'C:\SciSoft\Git\\theano_alexnet\scratch\ilsvrc12\
val_hkl'
# dir to store img_mean.npy, shuffled_train_filenames.npy,
train.txt, val.txt
misc_dir:'C:\SciSoft\Git\theano_alexnet\scratch\ilsvrc12\misc'
meta_clsloc_mat:'C:\SciSoft\Git\theano_alexnet\mnt\data\datasets\
ilsvrc_2014\ ILSVRC2014_devkit\data\meta_clsloc.mat'
val_label_file:'C:\SciSoft\Git\theano_alexnet\mnt\data\datasets\
ilsvrc_2014\ILSVRC2014_devkit\data\ILSVRC2014_clsloc_validation_
ground_truth.txt'
# training labels
valtxt_filename:'C:\SciSoft\Git\theano_alexnet\scratch\ilsvrc12\
misc\val.txt'
# validation labels
traintxt_filename:'C:\SciSoft\Git\theano_alexnet\scratch\ilsvrc12\
misc\train.txt'
```

In addition, in the file C:\SciSoft\Git\theano_alexnet\make_labels.py, add “import os.path” at the top of the file and replaced the line containing “filename = filename.split(’/’)[1]” with “filename = os.path.basename(filename)”. Also replace the line containing “key = train_filename.split(’/’)[-1]” with “key = os.path.basename(train_filename)”. These corrections are necessary because the Python .split delimiter “/” is not compatible with MS Windows path notations.

4.6.2 Preprocessed ImageNet Data for Theano-AlexNet

The 7 folders generated by running the shorter (~10 min) Python script (i.e., generate_toy_data.sh) to preprocess a subset of the ImageNet data for Theano-AlexNet are shown in Table 1.

Table 1 Folders generated in C:\SciSoft\Git\theano_alexnet\scratch\ilsvrc12

02/24/2016	02:43 PM	<DIR>	labels
02/23/2016	02:59 PM	<DIR>	misc
02/23/2016	05:25 PM	<DIR>	models
02/23/2016	02:58 PM	<DIR>	train_hkl_b256_b_128
02/23/2016	02:58 PM	<DIR>	train_hkl_b256_b_256
02/23/2016	02:59 PM	<DIR>	val_hkl_b256_b_128
02/23/2016	02:59 PM	<DIR>	val_hkl_b256_b_256

In the folder C:\SciSoft\Git\theano_alexnet\scratch\ilsvrc12\labels, the following 6 files are generated (Table 2).

Table 2 Files generated in C:\SciSoft\Git\theano_alexnet\scratch\ilsvrc12\labels

02/24/2016	02:43 PM	5,124,748	train_labels.npy
02/24/2016	02:43 PM	2,562,128	train_labels_0.npy
02/24/2016	02:43 PM	2,562,128	train_labels_1.npy
02/24/2016	02:39 PM	200,080	val_labels.npy
02/24/2016	02:43 PM	99,920	val_labels_0.npy
02/24/2016	02:43 PM	99,920	val_labels_1.npy

In the folder C:\SciSoft\Git\theano_alexnet\scratch\ilsvrc12\misc, the following 4 files are generated (Table 3).

Table 3 Files generated in C:\SciSoft\Git\theano_alexnet\scratch\ilsvrc12\misc

02/24/2016	02:38 PM	1,572,960	img_mean.npy
02/24/2016	02:54 PM	142,209,617	shuffled_train_filenames.npy
02/24/2016	02:39 PM	45,110,600	train.txt
02/24/2016	02:39 PM	1,694,500	val.txt

In the folders train_hkl_b256_b_256 and val_hkl_b256_b_256, 10 files (size = 50,333,799 each) are generated, which are labeled 0000.hkl through 0009.hkl. Note that each of these files contain 256 color images of size 256×256, hence 2,560 image files for training and validation are used for the short trial training of the Theano-AlexNet code. Figure 1 shows a few example images from this subset of the ILSVRC2012 data set, which illustrate time- and space-varying environmental conditions, such as variations in illumination, vegetation, terrain, and visibility.

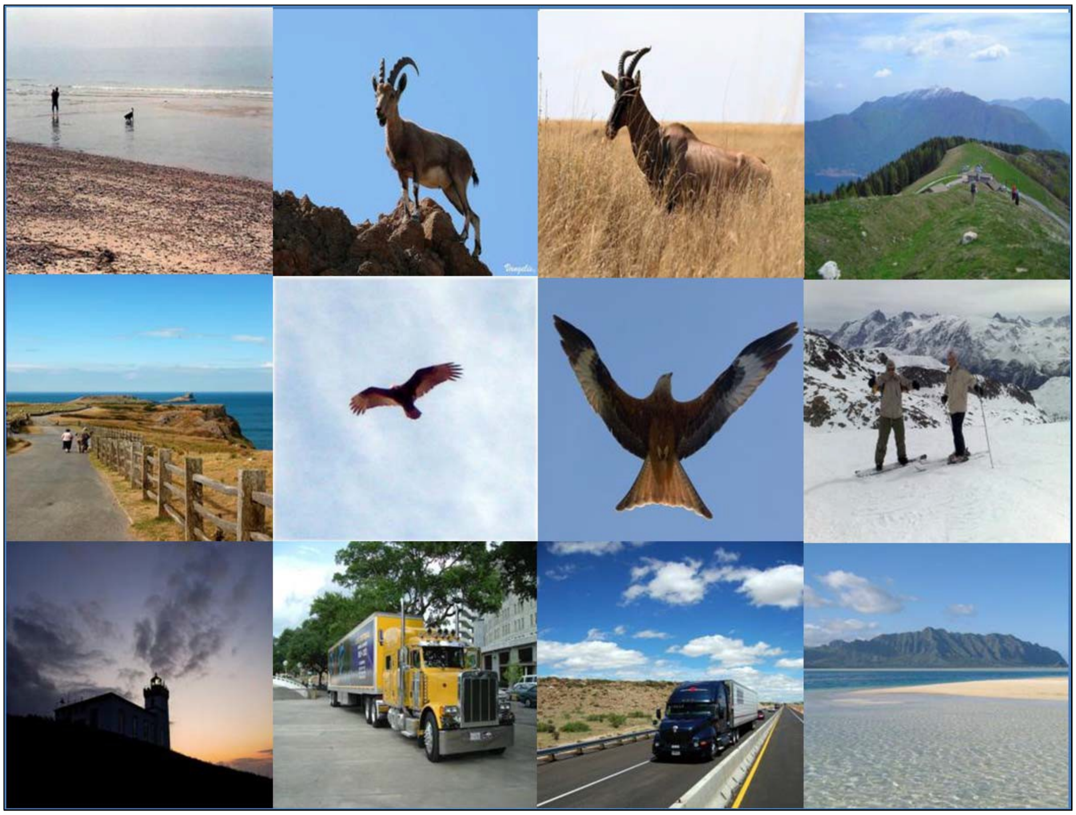

Fig. 1 A few example images from a subset of the ImageNet ILSVRC2012[43] data used for the short trial training of the Theano-AlexNet code that illustrate time- and space-varying environmental conditions in outdoor scenes, such as variations in illumination, vegetation, terrain, and visibility

4.7 Train Theano-AlexNet

Theano-AlexNet was tested using the file C:\SciSoft\Git\theano_alexnet\train.py as follows:

- C:\SciSoft\Git\theano_alexnet> python train.py THEANO_FLAGS=mode=FAST_RUN, floatX=float32.

In our first trial, Theano-AlexNet initialized properly (Table 4) and then executed 20,000 iterations in about 66 h, where upon the statement "Optimization complete" was returned to the command window (Table 5). Model output files from the last iteration were generated in the folder C:\SciSoft\Git\theano_alexnet\

scratch\ilsvrc12, which contained 11 weights and biases files, respectively, as well as 22 momentum files, all of which define the computations of the neural network.

Table 4 Building the model

conv (cudnn) layer with shape_in: (3, 227, 227, 256)
conv (cudnn) layer with shape_in: (96, 27, 27, 256)
conv (cudnn) layer with shape_in: (256, 13, 13, 256)
conv (cudnn) layer with shape_in: (384, 13, 13, 256)
conv (cudnn) layer with shape_in: (384, 13, 13, 256)
fc layer with num_in: 9216 num_out: 4096 dropout
layer with P_drop: 0.5
fc layer with num_in: 4096 num_out: 4096
dropout layer with P_drop: 0.5
softmax layer with num_in: 4096 num_out: 1000

Table 5 CNN training and validation results: 20,000 iterations

training @ iter = 20	***training @ iter = 19920***
training cost: 6.901418685916	training cost: 4.94013977051
training error rate: 1.0	training error rate: 0.95703125
validation loss: 6.907903	validation loss: 8.612438
validation error: 99.921875 %	validation error: 99.6875 %
training @ iter = 40	***training @ iter = 19940***
training cost: 6.89094781876	training cost: 5.020860672
training error rate: 1.0	training error rate: 0.91015625
validation loss: 6.907701	validation loss: 8.665205
validation error: 99.921875 %	validation error: 99.765625 %
training @ iter = 60	***training @ iter = 19960***
training cost: 6.88030338287	training cost: 4.81143093109
training error rate: 0.99609375	training error rate: 0.9375
validation loss: 6.907765	validation loss: 8.659591
validation error: 99.726562 %	validation error: 99.804688 %
training @ iter = 80	***training @ iter = 19980***
training cost: 6.87519598	training cost: 4.9219660759
training error rate: 1.0	training error rate: 0.93359375
validation loss: 6.908174	validation loss: 8.645909
validation error: 99.882812 %	validation error: 99.84375 %
...	***Optimization complete***

We found that with greater numbers of iterations the training cost and training error rates began to decrease. In fact, when we continued this model run, executing the code from 20,000 to 60,000 iterations over an additional 138 h, we found that the training cost at iteration = 60,000 was 0.0818 and the training error rate was 0.0273, which means that the CNN had “learned” to assign the correct class label to an image, when the image is taken from the training data set. However, the validation loss increased significantly (i.e., from 6.9079 to 26.7026) and the validation errors after 60,000 iterations remained high (i.e., 99.6484%), which indicates that the CNN is not assigning the correct class label to an image not previously seen, possibly due to overfitting.[41] For comparison, the validation error rates achieved by Ding et al.[42] after they ran the Theano-AlexNet model using 2 GPUs for 65 cycles were 42.6% for the top-1 class label and 19.9% for the top-5 class label. Thus, we recommend additional trials and analysis with increased numbers of training images in order to achieve lower validation error rates using the CNN so that we can better determine the feasibility of using the CNN to augment our approach, initially with simple single-object images and later on with more complicated scenes, such as those with time- and space-varying environmental conditions. As an example, Theano-AlexNet could be trained on the larger ImageNet[43] data set containing approximately million images, as described previously, even though this would require additional file storage (~500 Gb) for the input and output files and additional GPUs to achieve better computationally efficiency to implement the program code.

5. Summary and Conclusions

In this report, we outlined progress toward implementing our *mission driven scene understanding* approach to advance the value of Army autonomous intelligent systems. We described the proof-of-principle installation, setup, and testing of a convolutional neural network (CNN) program developed in Python and all of its required software dependencies. While we found that the CNN was able to determine the correct class labels for images taken from the training data set, the validation process did not appear to provide optimal results for images not previously seen. Thus, we recommend that additional trials and analysis be performed to better determine the feasibility of using the CNN to augment our approach, as described above. We anticipate that *mission driven scene understanding* will lead to 1) improved autonomous intelligent systems supporting Army missions in complex and changing environments and 2) improved course of action strategies based on scene understanding incorporating battlefield dynamic environments changing in space and time.

References

1. Army Research Laboratory (US). US Army Research Laboratory S&T Campaign Plans 2015–2035, Adelphi (MD): Army Research Laboratory (US); 2014. System Intelligence & Intelligent Systems; p. 98–99.
2. Greene MR, Oliva A. Recognition of natural scenes from global properties: seeing the forest without representing the trees. Cog Psych. 2009;58:137–176.
3. Greene MR, Oliva A. The briefest of glances: the time course of natural scene understanding. Psych Sci. 2009;20(4):464–472.
4. Itti L, Koch C, Niebur E. A model of saliency-based visual attention for rapid scene analysis. IEEE Trans Pattern Analysis and Machine Intel. 1998;20(11):1254–1259.
5. Perazzi F, Krahenbuhl P, Pritch Y, Hornung A. Saliency filters: contrast based filtering for salient region detection. Proceedings of IEEE Conference on Computer Vision and Pattern Recognition; 2012 Jun 16–21; Providence, RI.
6. Roberts R, Ta D-N, Straub J, Ok K, Dellaert F. Saliency detection and model-based tracking: a two part vision system for small robot navigation in forested environment. Proc. SPIE 8387, Unmanned Systems Technology XIV Conference; 2012 May 1. Bellingham (WA): The International Society for Optical Engineering; c2012.
7. Yeomans B, Shaukar A, Gao Y. Testing saliency based techniques for planetary surface scene analysis. In: ASTRA 2015. Proceedings of 13th Symposium on Advanced Space Technologies in Robotics and Automation; 2015 May 11–13; Noordwijk (The Netherlands).
8. Warnell G, David P, Chellappa R. Ray saliency: bottom-up visual saliency for a rotating and zooming camera. Intl J Computer Vis. 2015. doi 10.1007/s11263-015-0842-9.
9. Hung HSW. From visual saliency to video behaviour understanding [doctoral dissertation]. [London (UK)]: Queen Mary, University of London; 2007.
10. Meyers RE. Army Research Laboratory (US), Adelphi, MD. Personal communication, 2014.
11. Tunick A. Space and time scale characterization of image data in varying environmental conditions for better scene understanding. Adelphi (MD): Army Research Laboratory (US); Sep 2015. Report No.: ARL-TR-7419.

12. Tunick A. Space–time environmental image information for scene understanding. Adelphi (MD): Army Research Laboratory (US); 2016 Apr. Report No.: ARL-TR-7652.

13. Adelson EH. On seeing stuff: the perception of materials by humans and machines. Proc. SPIE 4299, Conference on Human Vision and Electronic Imaging VI Conference; 2001 Jan 20; San Jose (CA). Bellingham (WA): The International Society for Optical Engineering; c2001.

14. Siva P, Russell C, Xiang T, Agapito L. Looking beyond the image: Unsupervised learning for object saliency and detection. In: CVPR 2013. Proceedings of IEEE Conference on Computer Vision and Pattern Recognition; 2013 Jun 23–28; Portland (OR).

15. Battaglia PW, Hamrick JB, Tenenbaum JB. Simulation as an engine of physical scene understanding. Proc National Academy of Sciences. 2013;110(45):18327–18332.

16. Ullman TD, Stuhlmuller A, Goodman ND, Tenenbaum JB. Learning physics from dynamical scenes. In: CSS 2014. Proceedings of 36th Annual Conference of the Cognitive Science Society. 2014 Jul 23–26; Quebec City, Canada.

17. Divvala SK, Hoiem D, Hays JH, Efros AA, Hebert M. An empirical study of context in object detection. In: CVPR 2009. Proceedings of IEEE Conference on Computer Vision and Pattern Recognition; 2009 Jun 20–25; Miami (FL).

18. Rawat YS, Kankanhalli MS. Context-aware photography learning for smart mobile devices. ACM Trans Multimedia Computing, Comms, and Applications 2105 Oct;12(1s): Article 19.

19. Matas J, MurinoV, Leal-Taixe L, Rosenhahn B., editors. Holistic scene understanding. Report from the Dagstuhl Seminar 15081, 2015 Feb 15–20; Wadern, Germany.

20. Headquarters, Department of the Army. Weather support for Army tactical operations, appendix B – weather effects on Army operations. Washington (DC): Headquarters, Department of the Army; 1989 Aug 31. Field Manual No.: FM 34-81.

21. Headquarters, Department of the Army. Field behavior of NBC agents (including smoke and incendiaries). Washington (DC): Headquarters, Department of the Army; 1986 Nov 3. Field Manual No.: FM 3-6.

22. Headquarters, Department of the Army. Weather support and services for the US Army. Washington (DC): Headquarters, Department of the Army; 2015 Sep 10. Field Manual No.: AR 115-10.

23. Headquarters, Department of the Army. Army unmanned aircraft system operations. Washington (DC): Headquarters, Department of the Army; 2009 Jul 29. Field Manual No.: FM 3-04.155.

24. Datz IM. Military operations under special conditions of terrain and weather. New Delhi (India): Lancer Publishers; 2008.

25. Szymber RJ. US Army tactical weather support requirements for weather and environmental data elements and meteorological forecasts. Adelphi (MD): Army Research Laboratory (US); 2006 Feb. Report No.: ARL-TR-3720.

26. Karpathy A, Fei-Fei L. Deep visual-semantic alignments for generating image descriptions. arXiv:1412.2306v2; 2015.

27. Wigness M, Draper BA, Beveridge JR. Efficient label collection for unlabeled image datasets. In: CVPR 2015. Proceedings of IEEE Conference on Computer Vision and Pattern Recognition; 2015; Boston, MA. p. 4594–4602.

28. Wigness M, Robers III JG, Navarro-Sermenty LE, Suppey A, Draperz BA. Reducing adaptation latency for multi-concept visual perception in outdoor environments. In: IROS 2016. Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems; to be submitted. 2016 Oct 9–14; Daejeon, Korea.

29. Katsura H, Miura J, Hild M, Shirai Y. A view-based outdoor navigation using object recognition robust to changes of weather and seasons. In: IROS 2003. Proceedings of IEEE International Conference on Intelligent Robots and Systems; 2003; Las Vegas, NV. p. 2974–2979.

30. Milford MJ, Wyeth GF. SeqSLAM: Visual route-based navigation for sunny summer days and stormy winter nights. In: ICRA 2012. Proceedings of IEEE International Conference on Robotics and Automation. 2012; Saint Paul, MN.

31. Milford M. Vision-based place recognition: how low can you go? Intl J Robotics Res. 2013;32(7):766–789.

32. Milford M, Vig E, Scheirer W, Cox D. Vision-based simultaneous localization and mapping in changing outdoor environments. J Field Robotics. 2014;31(5):780–802.

33. Alvarez JM, Lopez AM, Gevers T, Lumbreras F. Combining priors, appearance and context for road detection. IEEE Trans Intelligent Transportation Sys. 2014;5(3):1168–1178.

34. Narasimhan SG, Nayar SK. Vision and the atmosphere. Intl J Computer Vision. 2002;48(3):233–254.

35. Narasimhan SG, Wang C, Nayar SK. All the images of an outdoor scene. In: Proceedings of 7th European Conference on Computer Vision; 2002; Copenhagen, Denmark. p. 148–162.

36. Lalonde J-F, Efros AA, Narasimhan SG. Estimating the natural illumination conditions from a single outdoor image. Intl J Computer Vis. 2012;98:123–145.

37. Xie L. Geographic and environmental interpretation of photographs [master's thesis]. Merced (CA): University of California, Merced; 2011.

38. Yu J, Luo J. Leveraging probabilistic season and location context models for scene understanding. In: Proceedings of the International Conference on Content-Based Image and Video Retrieval. 2008; Niagara Falls, Ontario (Canada). p. 169–178.

39. Bergstra J, Breuleux O, Bastien F, Lamblin P, Pascanu R, Desjardins G, Turian J, Warde-Farley D, Bengio Y. Theano: A CPU and GPU math expression compiler. In.: Proceedings of the Python for Scientific Computing Conference; 2010 Jun 30–Jul 3; Austin, TX.

40. Bastien F, Lamblin P, Pascanu R, Bergstra J, Goodfellow I, Bergeron A, Bouchard N, Warde-Farley D, Bengio Y. Theano: new features and speed improvements. Neural Information Processing Systems, Deep Learning Workshop; 2012 Dec 8; Lake Tahoe, CA.

41. Krizhevsky A, Sutskever I, Hinton GE. ImageNet classification with deep convolutional neural networks. Proc. Advances in Neural Info Processing Sys. 2012;25.

42. Ding W, Wang R, Mao F, Taylor G. Theano-based large-scale visual recognition with multiple GPUs. arXiv:1412.2302v4; 2015.

43. Russakovsky O, Deng J, Su H, Krause J, Satheesh S, Ma S, Huang Z, Karpathy A, Khosla A, Bernstien M, Berg AC, Fei-Fei L. ImageNet large scale visual recognition challenge. arXiv:1409.0575; 2014.

1 (PDF)	DEFENSE TECH INFO CTR DTIC OCA
2 (PDF)	US ARMY RSRCH LAB IMAL HRA MAIL & RECORDS MGMT RDRL CIO LL TECHL LIB
1 (PDF)	GOVT PRNTG OFC A MALHOTRA
8 (PDF)	US ARMY RESEARCH LABORATORY RDRL CII B BROOME M THOMAS RDRL CII A S YOUNG A TUNICK P DAVID G WARNELL RDRL CIN T R MEYERS K DEACON